# Exact and efficient calculation of Lagrange multipliers in biological polymers with constrained bond lengths and bond angles: Proteins and nucleic acids as example cases

Pablo García-Risueño*,[1,2,3], Pablo Echenique[1,2,3], and J. L. Alonso[3,2]

[1]*Instituto de Química Física Rocasolano, CSIC, Serrano 119, E-28006 Madrid, Spain*
[2]*Instituto de Biocomputación y Física de Sistemas Complejos (BIFI), Universidad de Zaragoza, Mariano Esquillor s/n, Edificio I+D, E-50018 Zaragoza, Spain*
[3]*Departamento de Física Teórica, Universidad de Zaragoza, Pedro Cerbuna 12, E-50009 Zaragoza, Spain*

June 21, 2011

**Abstract**

In order to accelerate molecular dynamics simulations, it is very common to impose holonomic constraints on the hardest degrees of freedom. In this way, the time step used to integrate the equations of motion can be increased, thereby, in principle, allowing longer total simulation times. The imposition of such constraints results in an aditional set of $N_c$ equations (the equations of constraint) and unknowns (their associated Lagrange multipliers), whose solution is closely related to any algorithm implementing the constraints in Euclidean coordinates. In this work it is shown that, due to the essentially linear structure of typical biological polymers, such as nucleic acids or proteins, the algebraic equations that need to be solved involve a matrix which is not only sparse, but also banded if the constraints are indexed in a skilful way. This allows the Lagrange multipliers to be obtained through a non-iterative procedure, which can be considered exact up to machine precision, and which takes $O(N_c)$ operations, instead of the usual $O(N_c^3)$ for generic molecular systems. We develop the formalism, and describe the appropriate indexing for a number of simple model molecules and also for alkanes, proteins and DNA. Finally, we provide a numerical example of the technique in a series of polyalanine peptides of different lengths using the AMBER force field. Although it is well known that a use of the Lagrange multipliers without any modification in the solution of the underlying ordinary differential equations yields unstable integration algorithms, the central role of these quantities makes their efficient calculation useful for the improvement of methods that correctly enforce the exact satisfaction of the constraints at each time step. We provide several examples of this.

*Email: garcia.risueno@gmail.com

**Keywords:** constraints, Lagrange multipliers, banded systems, molecular dynamics, proteins, DNA

# 1 Introduction

Due to the high frequency of the fastest internal motions in molecular systems, the discrete time step for molecular dynamics simulations must be very small (of the order of femtoseconds), while the actual span of biochemical processes typically require the choice of relatively long total times for simulations (e.g., from microseconds to milliseconds for protein folding processes). In addition, since biologically interesting molecules (such as proteins[1] and DNA[2]) consist of thousands of atoms, their trajectories in configuration space are esentially chaotic, and therefore reliable quantities can be obtained from the simulation only after statistical analysis[3]. In order to cope with these two requirements, which necessitate the computation of a large number of dynamic steps in order for predictions to be made, great effort is being made towards both hardware[4,5] and software[6,7] solutions. In fact, it is only very recently that simulations for interesting systems of hundreds of thousand of atoms in the millisecond scale are starting to become affordable, since, as already mentioned, the main limitation of these computational techniques is the large difference between the elemental time step used to integrate the equations of motion and the total time span needed to obtain useful information. In this context, strategies to increase the time step are very valuable.

A widely used method to this end is to constrain some of the internal degrees of freedom[8] of a molecule (typically bond lengths, sometimes bond angles and rarely dihedral angles). For a Verlet-like integrator[9,10], stability requires the time step to be at least about five times smaller than the period of the fastest vibration in the studied system[11]. Here is where constraints come into play. By constraining the hardest degrees of freedom, the fastest vibrational motions are frozen, and thus larger time steps still produce stable simulations. If constraints are imposed on bond lengths involving hydrogens, the time step can typically be increased by a factor of 2 to 3 (from 1 fs to 2 or 3 fs)[12]. Constraining additional internal degrees of freedom, such as heavy atoms bond lengths and bond angles, allows even larger timesteps[11,13], but one has to be careful since, as more and softer degrees of freedom are constrained, the more likely it is that the physical properties of the simulated system could be severely distorted[14–16].

Essential ingredients in the calculation of the forces produced by the imposition of constraints are the so-called Lagrange multipliers[17], and their efficient numerical evaluation (or that of related quantities, as modified Lagrange multipliers which avoid any drift on constraints) is therefore of the utmost importance. In this work, we show that the fact that many interesting biological molecules are esentially linear polymers allows the Lagrange multipliers to be calculated in order $N_c$ operations (for a molecule where $N_c$ constraints are imposed) in an exact (up to machine precision), non-iterative way. Moreover, we provide a method to do so which is based on a skilful ordering of the constraints indices, and in a recently introduced algorithm for solving linear banded systems[18].

There exist some previous works which comment that solving this kind of linear problems (or related ones) is costly (but without giving further details)[19–22], and some other

works explicitly state that such a computation must take $O(N_c^3)$[23] or $O(N_c^2)$[24,25] operations. We prove here that this can be done in $O(N_c)$ steps.

In a more related family of works, linear algebra techniques for sparse systems have been used to tackle the problem of constrained molecular dynamics in the same spirit as this work. The most prominent examples[16] use sparse linear algebra codes (SPARSPAK and MA28) to achieve more efficient matrix factorizations. They use a matrix equation similar to the one appearing in SHAKE[19], which is derived from the equation $\sigma(t + \Delta t) = 0$. Then, the sparse algebra algorithms reorder the rows and columns of this matrix in a way which makes Gaussian elimination efficient (i.e., which generates few fill-ins). Simpler sparse techniques are used in other methods, as those in[12,24,26–28]. In the constant matrix approximation introduced in[12], small value entries of the inverse of a given sparse matrix are neglected, and this sparse approximation to the inverse is used in an iterative procedure in the search for modified Lagrange Multipliers which satisfy the constraints without any drift. In P-SHAKE[24], the sparseness of some different but related matrices is also used to accelerate the computation of matrix-vector products with the same aim. In a parallel implementation[26] of the well-known SHAKE method[19], a conjugate gradient minimization step is made efficient due, again, to sparse matrix-vector products. Coming to not only sparse systems but banded ones, we can mention the work by Mazars[29], who showed that the Lagrange multipliers can be computed analytically for the simple linear chain by inverting a tridiagonal matrix (a particular case of the more general calculations introduced in the present paper). The method known as MILC-SHAKE[27], takes the efficient inversion of this tridiagonal matrix to the practical arena, introducing a method to implement constraints based on SHAKE. However, this method, as the calculations in ref.[29], is only applicable to a very small family of systems (linear chains and rings). MILCH-SHAKE[28] neglects small entries in the coordinate matrix to invert so that it is tridiagonal, and uses its solution as starting guess for an iterative procedure. Also, in ref.[30], the overdamped Langevin dynamics of a molecular system is considered, and a banded matrix appears which involves the constraints as well as the friction. This matrix, which is different from the one discussed in this work, can also be inverted using similar techniques. It is also worth mentioning that the idea of avoiding operating on zeros to reduce the scaling of the solution of sparse systems has a long history and has been used in other contexts. For example, in ref.[31], the "nested disection" method is introduced. This approach is related to the solution of a linear system related to a finite-differences problem in a regular mesh, and the matrix $A$ to invert is $N \times N$, with $N = (n + 1)^2$, being $n$ the size of the mesh. Since each row/column of this matrix contains approximately $n \simeq N^{1/2}$ non-zero elements, the solution of the associated linear system needs of the order of $n^3$ operations, or $N^{3/2}$, which is harsher than the case of a linear polymer presented in our work, in which we have an $N$-independent number of non-zero elements per row/column, and hence can achieve order $N$ scaling. Also, a theorem by Rose[32] mentioned in this work indicates that our approach is close to optimal for the given problem we are dealing with. Finally, in the field of robot kinematics, many $O(N_c)$ algorithms have been devised to deal with different aspects of constrained physical systems (robots in this case)[33–35], but none of them tackles the exact calculation of the Lagrange multipliers themselves.

In summary, despite these previous developments along similar lines to the one followed in this work, as far as we are aware, the method introduced here represents the first time in which the matrix to invert in order to find the Lagrange multipliers is explicitly

constructed for a general biological polymer, in such a way that it is not only sparse but also banded, therefore allowing to solve the associated linear problem in just $O(N_c)$ steps.

This work is structured as follows. In sec. 2, we introduce the basic formalism for the calculation of constraint forces and Lagrange multipliers. In sec. 3, we apply the introduced technique to a series of polyalanine peptides using the AMBER force field, comparing the relative efficiency between the calculation of the Lagrange multipliers in the traditional way ($O(N_c^3)$) and in the new way presented here ($O(N_c)$). It is also worth pointing out that it is well-known that the use of the exact Lagrange multipliers (the ones computed by the new method discussed in this work) to solve the underlying ordinary differential equations behind the differential-algebraic problem introduced in sec. 2 produces unstable algorithms in which the system abandons the constrained subspace [19,30,36]. However, the Lagrange multipliers and the matrix that needs to be inverted to find them are quantities that are intrincately related with every formalism dealing with constraints, therefore participating of the methods that *do* solve the problem correctly, i.e., exactly enforcing the constraints at each time step [19,37]. This makes the calculations presented here potentially very useful for the improvement of many of these methods. In sec. 4, we not only summarize the main conclusions of this work, but we also outline a number of examples in which the new technique can be applied in practical algorithms.

Finally, note that, in the supplementary material, we explain how to index the constraints in order for the resulting linear system of equations to be banded with the minimal bandwidth (which is essential to solve it efficiently). We do this by starting with very simple toy systems and building on complexity as we move forward towards the final discussion about DNA and proteins; this way of proceeding is intended to help the reader build the corresponding indexing for molecules not covered in this work.

## 2 Calculation of the Lagrange multipliers

If holonomic, rheonomous constraints are imposed on a classical system of $n$ atoms, and the D'Alembert's principle is assumed to hold, its motion is the solution of the following system of differential equations [17,38]:

$$m_\alpha \frac{\mathrm{d}^2 \vec{x}_\alpha(t)}{\mathrm{d}t^2} = \vec{F}_\alpha(\mathbf{x}(t)) + \sum_{I=1}^{N_c} \lambda_I(t) \vec{\nabla}_\alpha \sigma^I(\mathbf{x}(t)) \,, \qquad \alpha = 1, \ldots, n \,, \tag{2.1a}$$

$$\sigma^I(\mathbf{x}(t)) = 0 \,, \qquad I = 1, \ldots, N_c \,, \tag{2.1b}$$

$$\mathbf{x}(t_0) = \mathbf{x}_0 \,, \tag{2.1c}$$

$$\frac{\mathrm{d}\mathbf{x}(t_0)}{\mathrm{d}t} = \dot{\mathbf{x}}_0 \,, \tag{2.1d}$$

where (2.1a) is the modified Newton's second law and (2.1b) are the equations of the constraints themselves; $\lambda_I$ are the Lagrange multipliers associated with the constraints; $\vec{F}_\alpha$ represents the external force acting on atom $\alpha$, $\vec{x}_\alpha$ is its Euclidean position, and $\mathbf{x}$ collectively denote the set of all such coordinates. We assume $\vec{F}_\alpha$ to be conservative, i.e., to come from the gradient of a scalar potential function $V(x)$; and $\sum_{I=1}^{N_c} \lambda_I \vec{\nabla}_\alpha \sigma^I(\mathbf{x})$ should be regarded as the *force of constraint* acting on atom $\alpha$.

Also, in the above expression and in this whole document we will use the following notation for the different indices:

- $\alpha, \beta, \gamma, \epsilon, \zeta = 1, \ldots, n$ (except if otherwise stated) for atoms.
- $\mu, \nu = 1, \ldots, 3n$ (except if otherwise stated) for the atoms coordinates when no explicit reference to the atom index needs to be made.
- $I, J = 1, \ldots, N_c$ for constraints and the rows and columns of the associated matrices.
- $k, l$ as generic indices for products and sums.

The existence of $N_c$ constraints turns a system of $N = 3n$ differential equations with $N$ unknowns into a system of $N + N_c$ algebraic-differential equations with $N + N_c$ unknowns. The constraints equations in (2.1b) are the new equations, and the Lagrange multipliers are the new unknowns whose value must be found in order to solve the system.

If the functions $\sigma^I(\mathbf{x})$ are analytical, the system of equations in (2.1) is equivalent to the following one:

$$m_\alpha \frac{\mathrm{d}^2 \vec{x}_\alpha(t)}{\mathrm{d}t^2} = \vec{F}_\alpha(\mathbf{x}(t)) + \sum_{I=1}^{N_c} \lambda_I(t) \vec{\nabla}_\alpha \sigma^I(\mathbf{x}(t)) \,, \tag{2.2a}$$

$$\sigma^I(\mathbf{x}(t_0)) = 0 \,, \tag{2.2b}$$

$$\frac{\mathrm{d}\sigma^I(\mathbf{x}(t_0))}{\mathrm{d}t} = 0 \,, \tag{2.2c}$$

$$\frac{\mathrm{d}^2\sigma^I(\mathbf{x}(t))}{\mathrm{d}t^2} = 0 \,, \quad \forall t \,, \tag{2.2d}$$

$$\mathbf{x}(t_0) = \mathbf{x}_0 \,, \tag{2.2e}$$

$$\frac{\mathrm{d}\mathbf{x}(t_0)}{\mathrm{d}t} = \dot{\mathbf{x}}_0 \,. \tag{2.2f}$$

In this new form, there exists a more direct path for the solution of the Lagrange multipliers: If we explicitly calculate the second derivative in eq. (2.2d) and then substitute eq. (2.2a) where the accelerations appear, we arrive at

$$\begin{aligned} \frac{\mathrm{d}^2\sigma^I}{\mathrm{d}t^2} &= \sum_\mu \frac{1}{m_\mu} \left( F_\mu + \sum_J \lambda_J \frac{\partial \sigma^J}{\partial x^\mu} \right) \frac{\partial \sigma^I}{\partial x^\mu} + \sum_{\mu,\nu} \frac{\mathrm{d}x^\mu}{\mathrm{d}t} \frac{\mathrm{d}x^\nu}{\mathrm{d}t} \frac{\partial^2 \sigma^I}{\partial x^\mu \partial x^\nu} \\ &:= p^I + q^I + \sum_J R_{IJ} \lambda_J = 0 \,, \qquad I = 1, \ldots, N_c \,, \end{aligned} \tag{2.3}$$

where we have implicitly defined

$$p^I := \sum_\mu \frac{1}{m_\mu} F_\mu \frac{\partial \sigma^I}{\partial x^\mu} = \sum_\alpha \frac{1}{m_\alpha} \vec{F}_\alpha \cdot \vec{\nabla}_\alpha \sigma^I \,, \tag{2.4a}$$

$$q^I := \sum_{\mu,\nu} \frac{\mathrm{d}x^\mu}{\mathrm{d}t} \frac{\mathrm{d}x^\nu}{\mathrm{d}t} \frac{\partial^2 \sigma^I}{\partial x^\mu \partial x^\nu} \,, \tag{2.4b}$$

$$R_{IJ} := \sum_\mu \frac{1}{m_\mu} \frac{\partial \sigma^I}{\partial x^\mu} \frac{\partial \sigma^J}{\partial x^\mu} = \sum_\alpha \frac{1}{m_\alpha} \vec{\nabla}_\alpha \sigma^I \cdot \vec{\nabla}_\alpha \sigma^J \,, \tag{2.4c}$$

and it becomes clear that, at each $t$, the Lagrange multipliers $\lambda_J$ are actually a *known* function of the positions and the velocities.

We shall use the shorthand

$$o^I := p^I + q^I \,, \qquad I = 1, \ldots, N_c \,, \tag{2.5}$$

and, $o$, $p$, and $q$ to denote the whole $N_c$-tuples, as usual.

Now, in order to obtain the Lagrange multipliers $\lambda_J$, we just need to solve

$$\sum_I R_{IJ}\lambda_J = -\left(p^I + q^I\right) \quad \Rightarrow \quad R\lambda = -o \,. \tag{2.6}$$

This is a linear system of $N_c$ equations and $N_c$ unknowns. In the following, we will prove that the solution to it, when constraints are imposed on typical biological polymers, can be found in $O(N_c)$ operations without the use of any iterative or truncation procedure, i.e., in an exact way up to machine precision. To show this, first we will prove that the value of the vectors $p$ and $q$ can be obtained in $O(N_c)$ operations. Then, we will show that the same is true for all the non-zero entries of matrix $R$, and finally we will briefly discuss the results in[18], where we introduced an algorithm to solve the system in (2.6) also in $O(N_c)$ operations.

It is worth remarking at this point that, in this work, we will only consider constraints that hold the distance between pairs of atoms constant, i.e.,

$$\sigma^{I(\alpha,\beta)}(\mathbf{x}) := |\vec{x_\alpha} - \vec{x_\beta}|^2 - (a_{\alpha,\beta})^2 \,, \tag{2.7}$$

where $a_{\alpha,\beta}$ is a constant number, and the fact that we can establish a correspondence between constrained pairs $(\alpha,\beta)$ and the constraints indices has been explicitly indicated by the notation $I(\alpha,\beta)$.

This can represent a constraint on:

- a bond length between atoms $\alpha$ and $\beta$,
- a bond angle between atoms $\alpha$, $\beta$ and $\gamma$, if both $\alpha$ and $\beta$ are connected to $\gamma$ through constrained bond lengths,
- a principal dihedral angle involving $\alpha$, $\beta$, $\gamma$ and $\delta$ (see[39] for a rigorous definition of the different types of internal coordinates), if the bond lengths $(\alpha,\beta)$, $(\beta,\gamma)$ and $(\gamma,\delta)$ are constrained, as well as the bond angles $(\alpha,\beta,\gamma)$ and $(\beta,\gamma,\delta)$,
- or a phase dihedral angle involving $\alpha$, $\beta$, $\gamma$ and $\delta$ if the bond lengths $(\alpha,\beta)$, $(\beta,\gamma)$ and $(\beta,\delta)$ are constrained, as well as the bond angles $(\alpha,\beta,\gamma)$ and $(\alpha,\beta,\delta)$.

This way of constraining degrees of freedom is called *triangularization*. If no triangularization is desired (as, for example, if we want to constrain dihedral angles but not bond angles), different explicit expressions from those in the following paragraphs must be written down, but the basic concepts introduced here are equally valid and the main conclusions still hold. If the constrains are complicated functions of the atomic positions (not simple lengths) but still involve each one of them a small number of atoms (such as the cosine of a dihedral angle), we can see from the definition of matrix $R$ in eq. (2.4c) that the associated linear system will be again sparse.

Now, from eq. (2.7), we obtain

$$\vec{\nabla}_\gamma \sigma^{I(\alpha,\beta)} = 2(\vec{x_\alpha} - \vec{x_\beta})(\delta_{\gamma,\alpha} - \delta_{\gamma,\beta}) \,. \tag{2.8}$$

where $\delta$ represents the Kroenecker delta. Inserting this into (2.4a), we get a simple expression for $p^{I(\alpha,\beta)}$

$$\begin{aligned} p^{I(\alpha,\beta)} &:= \sum_\mu \frac{1}{m_\mu} F_\mu \frac{\partial \sigma^{I(\alpha,\beta)}}{\partial x^\mu} = \sum_\gamma \frac{1}{m_\gamma} \vec{F}_\gamma \cdot \vec{\nabla}_\gamma \sigma^{I(\alpha,\beta)} \\ &= \sum_\gamma \frac{2}{m_\gamma} \vec{F}_\gamma \cdot (\vec{x_\alpha} - \vec{x_\beta})(\delta_{\gamma,\alpha} - \delta_{\gamma,\beta}) = 2(\vec{x_\alpha} - \vec{x_\beta}) \cdot \left( \frac{\vec{F}_\alpha}{m_\alpha} - \frac{\vec{F}_\beta}{m_\beta} \right) . \end{aligned} \tag{2.9}$$

The calculation of $q^{I(\alpha,\beta)}$ is more involved, but it also resolves into a simple expression: First, we remember that the indices run as $\mu, \nu = 1, \ldots, 3n$, and $\alpha = 1, \ldots, n$, and we produce the following trivial relationship:

$$\begin{aligned} \vec{x}_\alpha &= x^{3\alpha-2}\hat{i} + x^{3\alpha-1}\hat{j} + x^{3\alpha}\hat{k} \\ \Rightarrow \quad \frac{\partial \vec{x}_\alpha}{\partial x^\mu} &= \frac{\partial (x^{3\alpha-2}\hat{i} + x^{3\alpha-1}\hat{j} + x^{3\alpha}\hat{k})}{\partial x^\mu} = \delta_{3\alpha-2,\mu}\hat{i} + \delta_{3\alpha-1,\mu}\hat{j} + \delta_{3\alpha,\mu}\hat{k} \,, \end{aligned} \tag{2.10}$$

where $\hat{i}$, $\hat{j}$ and $\hat{k}$ are the unitary vectors along the $x$, $y$ and $z$ axes, respectively.

Therefore, we can compute the first derivative of $\sigma^{I(\alpha,\beta)}$:

$$\begin{aligned} \frac{\partial \sigma^{I(\alpha,\beta)}}{\partial x^\mu} &= \frac{\partial ((\vec{x}_\alpha - \vec{x}_\beta)^2 - a^2_{\alpha,\beta})}{\partial x^\mu} \\ &= 2(\vec{x}_\alpha - \vec{x}_\beta) \cdot [(\delta_{3\alpha-2,\mu}\hat{i} + \delta_{3\alpha-1,\mu}\hat{j} + \delta_{3\alpha,\mu}\hat{k}) \\ &\qquad -(\delta_{3\beta-2,\mu}\hat{i} + \delta_{3\beta-1,\mu}\hat{j} + \delta_{3\beta,\mu}\hat{k})] \,, \end{aligned} \tag{2.11}$$

and also the second derivative:

$$\begin{aligned} \frac{\partial^2 \sigma^{I(\alpha,\beta)}}{\partial x^\mu \partial x^\nu} &= 2[(\delta_{3\alpha-2,\mu}\hat{i} + \delta_{3\alpha-1,\mu}\hat{j} + \delta_{3\alpha,\mu}\hat{k}) - (\delta_{3\beta-2,\mu}\hat{i} + \delta_{3\beta-1,\mu}\hat{j} + \delta_{3\beta,\mu}\hat{k})] \\ &\quad \cdot [(\delta_{3\alpha-2,\nu}\hat{i} + \delta_{3\alpha-1,\nu}\hat{j} + \delta_{3\alpha,\nu}\hat{k}) - (\delta_{3\beta-2,\nu}\hat{i} + \delta_{3\beta-1,\nu}\hat{j} + \delta_{3\beta,\nu}\hat{k})] \\ &= 2(\delta_{3\alpha-2,\mu}\delta_{3\alpha-2,\nu} + \delta_{3\beta-2,\mu}\delta_{3\beta-2,\nu} - \delta_{3\alpha-2,\mu}\delta_{3\beta-2,\nu} - \delta_{3\beta-2,\mu}\delta_{3\alpha-2,\nu} \\ &\quad + \delta_{3\alpha-1,\mu}\delta_{3\alpha-1,\nu} + \delta_{3\beta-1,\mu}\delta_{3\beta-1,\nu} - \delta_{3\alpha-1,\mu}\delta_{3\beta-1,\nu} - \delta_{3\beta-1,\mu}\delta_{3\alpha-1,\nu} \\ &\quad + \delta_{3\alpha,\mu}\delta_{3\alpha,\nu} + \delta_{3\beta,\mu}\delta_{3\beta,\nu} - \delta_{3\alpha,\mu}\delta_{3\beta,\nu} - \delta_{3\beta,\mu}\delta_{3\alpha,\nu}) \,. \end{aligned} \tag{2.12}$$

Taking this into the original expression for $q^{I(\alpha,\beta)}$ in eq. (2.4b) and playing with the

sums and the deltas, we arrive at

$$\begin{aligned} q^{I(\alpha,\beta)} &:= \sum_{\mu,\nu} \frac{\mathrm{d}x^\mu}{\mathrm{d}t}\frac{\mathrm{d}x^\nu}{\mathrm{d}t}\frac{\partial^2\sigma^{I(\alpha,\beta)}}{\partial x^\mu \partial x^\nu} \\ &= 2\left(\frac{\mathrm{d}x^{3\alpha-2}}{\mathrm{d}t}\right)^2 + 2\left(\frac{\mathrm{d}x^{3\beta-2}}{\mathrm{d}t}\right)^2 - 4\left(\frac{\mathrm{d}x^{3\alpha-2}}{\mathrm{d}t}\frac{dx^{3\beta-2}}{dt}\right) \\ &\quad + 2\left(\frac{\mathrm{d}x^{3\alpha-1}}{\mathrm{d}t}\right)^2 + 2\left(\frac{\mathrm{d}x^{3\beta-1}}{\mathrm{d}t}\right)^2 - 4\left(\frac{\mathrm{d}x^{3\alpha-1}}{\mathrm{d}t}\frac{\mathrm{d}x^{3\beta-1}}{\mathrm{d}t}\right) \\ &\quad + 2\left(\frac{\mathrm{d}x^{3\alpha}}{\mathrm{d}t}\right)^2 + 2\left(\frac{\mathrm{d}x^{3\beta}}{\mathrm{d}t}\right)^2 - 4\left(\frac{\mathrm{d}x^{3\alpha}}{\mathrm{d}t}\frac{\mathrm{d}x^{3\beta}}{\mathrm{d}t}\right) \\ &= 2\left|\frac{\mathrm{d}\vec{x}_\alpha}{\mathrm{d}t} - \frac{\mathrm{d}\vec{x}_\beta}{\mathrm{d}t}\right|^2 . \end{aligned} \tag{2.13}$$

Now, eqs. (2.5), (2.9) and (2.13) can be gathered together to become

$$o^{I(\alpha,\beta)} = 2\left|\frac{\mathrm{d}\vec{x}_\alpha}{\mathrm{d}t} - \frac{\mathrm{d}\vec{x}_\beta}{\mathrm{d}t}\right|^2 + 2(\vec{x_\alpha} - \vec{x}_\beta)\cdot\left(\frac{\vec{F}_\alpha}{m_\alpha} - \frac{\vec{F}_\beta}{m_\beta}\right) , \tag{2.14}$$

where we can see that the calculation of $o^{I(\alpha,\beta)}$ always takes the same number of operations, independently of the number of atoms in our system, $n$, and the number of constraints imposed on it, $N_c$. Therefore, calculating the whole vector $o$ in eq. (2.6) scales like $N_c$.

In order to obtain an explicit expression for the entries of the matrix $R$, we now introduce eq. (2.8) into its definition in eq. (2.4c):

$$\begin{aligned} R_{I(\alpha,\beta),J(\gamma,\epsilon)} &:= \sum_{\zeta=1}^{n} \frac{1}{m_\zeta}\vec{\nabla}_\zeta\sigma^{I(\alpha,\beta)}\cdot\vec{\nabla}_\zeta\sigma^{J(\gamma,\epsilon)} \\ &= \sum_{\zeta=1}^{n} \frac{4}{m_\zeta}(\vec{x}_\alpha - \vec{x}_\beta)\cdot(\vec{x}_\gamma - \vec{x}_\epsilon)(\delta_{\zeta,\alpha} - \delta_{\zeta,\beta})(\delta_{\zeta,\gamma} - \delta_{\zeta,\epsilon}) \\ &= 4(\vec{x}_\alpha - \vec{x}_\beta)\cdot(\vec{x}_\gamma - \vec{x}_\epsilon)\left(\frac{\delta_{\alpha,\gamma}}{m_\alpha} - \frac{\delta_{\alpha,\epsilon}}{m_\alpha} - \frac{\delta_{\beta,\gamma}}{m_\beta} + \frac{\delta_{\beta,\epsilon}}{m_\beta}\right) , \end{aligned} \tag{2.15}$$

where we have used that

$$\sum_{\zeta=1}^{n} \delta_{\zeta,\alpha}\delta_{\zeta,\beta} = \delta_{\alpha,\beta} . \tag{2.16}$$

Looking at this expression, we can see that a constant number of operations (independent of $n$ and $N_c$) is required to obtain the value of every entry in $R$. The terms proportional to the Kroenecker deltas imply that, as we will see later, in a typical biological polymer, the matrix $R$ will be sparse (actually banded if the constraints are appropriately ordered as we describe in the following sections), being the number of non-zero entries actually proportional to $N_c$. More precisely, the entry $R_{IJ}$ will only be non-zero if the constraints $I$ and $J$ share an atom.

Now, since both the vector $o$ and the matrix $R$ in eq. (2.6) can be computed in $O(N_c)$ operations, it only remains to be proved that the solution of the linear system of equations

is also an $O(N_c)$ process, but this is a well-known fact when the matrix defining the system is banded. In[18], we introduced a new algorithm to solve this kind of banded systems which is faster and more accurate than existing alternatives. Essentially, we showed that the linear system of equations

$$Ax = b\,, \tag{2.17}$$

where $A$ is a $d \times d$ matrix, $x$ is the $d \times 1$ vector of the unknowns, $b$ is a given $d \times 1$ vector and $A$ is *banded*, i.e., it satisfies that for known $m < n$

$$A_{I,I+K} = 0 \quad \forall\, K > m\,, \forall I\,, \tag{2.18}$$
$$A_{I+L,I} = 0 \quad \forall\, L > m\,, \forall I\,, \tag{2.19}$$

can be directly solved up to machine precision in $O(d)$ operations.

This can be done using the following set of recursive equations for the auxiliary quantities $\xi_{IJ}$:

$$\xi_{II} = \left(A_{II} - \sum_{M=max(1,I-m)}^{I-1} \xi_{IM}\xi_{MI}\right)^{-1}\,, \tag{2.20a}$$

$$\xi_{IJ} = \xi_{II}\left(-A_{IJ} + \sum_{M=\max\{1,J-m\}}^{I-1} \xi_{IM}\xi_{MJ}\right)\,, \qquad \text{for} \quad I < J\,, \tag{2.20b}$$

$$\xi_{IJ} = -A_{IJ} + \sum_{M=\max\{1,I-m\}}^{J-1} \xi_{IM}\xi_{MJ}\,, \qquad \text{for} \quad I > J\,, \tag{2.20c}$$

$$c_I = b_I + \sum_{M=\max\{I-m,1\}}^{I-1} \xi_{IM}c_M\,, \tag{2.20d}$$

$$x_I = \xi_{II}c_I + \sum_{K=I+1}^{\min\{I+m,n\}} \xi_{IK}x_K\,. \tag{2.20e}$$

In (2.20), the $\xi$ coefficients are related to the process of Gaussian elimination (GE), which is used to solve linear systems of equations by adding their equations multiplied by appropriate terms. In GE, the matrix $A$ in (2.17) is gradually convered to the identity matrix, while the independent term $b$ is consistently modified. In ref.[18] it is explicitly explained how to derive (2.20).

If the matrix $A$ is symmetric ($A_{IJ} = A_{JI}$), as it is the case with $R$ [see (2.4c)], we can additionally save about one half of the required operations just by using

$$\xi_{IJ} = \xi_{JI}/\xi_{JJ}\,, \qquad \text{for} \quad I > J\,, \tag{2.21}$$

instead of (2.20c). Eq. (2.21) can be obtained from (2.20) by induction. In GE, the coefficients $\xi_{II}$ are not univocally determined, so their exact expression is to be chosen among infinite proportional ones. We recommend the form given in (2.20) for the $\xi$ coefficients because other valid ones (like, for example, considering $\xi_{IJ} = \xi_{JI}$, $\xi_{II} = 1/\sqrt{A_{II} - \sum_{M=max(1,I-m)}^{I-1} \xi_{IM}\xi_{MI}}$, which involves square roots) are computationally more expensive.

In the supplementary material, we show how to index the constraints in such a way that nearby indices correspond to constraints where involved atoms are close to each other and likely participate of the same constraints. In such a case, not only will the matrix $R$ in eq. (2.6) be banded, allowing the use of the method described above, but it will also have a minimal bandwidth $m$, which is also an important point, since the computational cost for solving the linear system scales as $O(N_c m^2)$ (when the bandwidth is constant).

# 3 Numerical calculations

In this section, we apply the efficient technique introduced in this work to a series of polyalanine molecules in order to calculate the Lagrange multipliers when bond length constraints are imposed. We also compare our method, both in terms of accuracy and numerical efficiency, to the traditional inversion of the matrix $R$ without taking into account its banded structure. Notice that the example in this section involves constraints only in bond lengths for simplicity, but the whole formalism can be straightforwardly applied to bond angle constraints, as it is clear from the discussion in the previous section and the examples in the supplementary material.

We used the code Avogadro[40] to build polyalanine chains of $N_{\text{res}}$ =2, 5, 12, 20, 30, 40, 50, 60, 80, 90 and 100 residues, and we chose their initial conformation to be approximately an alpha helix, i.e., with the values of the Ramachandran angles in the backbone $\phi = -60^o$ and $\psi = -40^o$ [1]. Next, for each of these chains, we used the molecular dynamics package AMBER[41] to produce the atoms positions ($\mathbf{x}$), velocities ($\mathbf{v}$) and external forces ($\mathbf{F}$) needed to calculate the Lagrange multipliers (see sec. 2) after a short equilibration molecular dynamics simulations. We chose to constrain all bond lengths, but our method is equally valid for any other choice, as the more common constraining only of bonds that involve hydrogens.

In order to produce reasonable final conformations, we repeated the following process for each of the chains:

- Solvation with explicit water molecules.
- Minimization of the solvent positions holding the polypeptide chain fixed (3,000 steps).
- Minimization of all atoms positions (3,000 steps), without constraints.
- Thermalization: changing the temperature from 0 K to 300 K during 10,000 molecular dynamics steps. With constraints.
- Stabilization: 20,000 molecular dynamics steps at a constant temperature of 300 K. With constraints.
- Measurement of $x$, $v$ and $F$.

Neutralization is not necessary, because our polyalanine chains are themselves neutral. In all calculations we used the force field described in[42], chose a cutoff for Coulomb interactions of 10 Å and a time step equal to 0.002 ps, and imposed constraints on all

bond lengths as mentioned. In the thermostated steps, we used Langevin dynamics with a collision frequency of 1 $ps^{-1}$.

Using the information obtained and the indexing of the constraints described in this work, we constructed the matrix $R$ and the vector $o$ and proceeded to find the Lagrange multipliers using eq. (2.6). Since (2.6) is a linear problem, one straightforward way to solve it is to use traditional Gauss-Jordan elimination or LU factorization[17,43]. But these methods have a drawback: they scale with the cube of the size of the system, i.e., if we imposed $N_c$ constraints on our system (and therefore we needed to obtain $N_c$ Lagrange multipliers), the number of floating point operations that these methods would require is proportional to $N_c^3$. However, as we showed in the previous sections, the fact that many biological molecules, and proteins in particular, are essentially linear, allows the constraints to be indexed in such a way that the matrix $R$ in eq. (2.6) is banded, and different techniques to be used for solving the problem which require only $O(N_c)$ floating point operations[18].

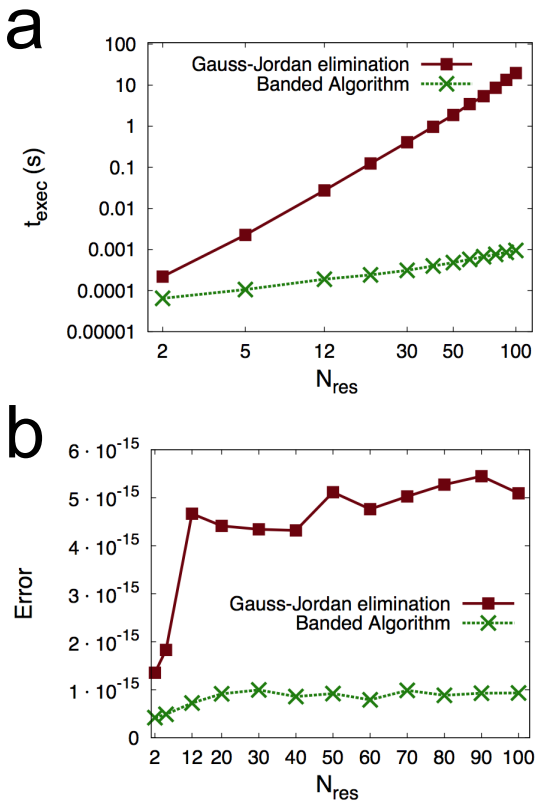


Figure 3.1: Comparison of **a)** numerical complexity (measured as execution time) and **b)** accuracy (given by (3.1)) between a traditional Gauss-Jordan solver (solid line) and the banded algorithm described in this work (dashed line), for the calculation of the Lagrange multipliers on a series of polyalanine chains as a function of their number of residues $N_{res}$. Note that the slopes in **a)** are approximately 1 and 3, since the slopes in log vs. log plots are the scaling exponent.

| $N_{\text{res}}$ | Gauss-Jordan Error | Banded Alg. Error | Gauss-Jordan $t_{\text{exec}}$ (s) | Banded Alg. $t_{\text{exec}}$ (s) |
|---|---|---|---|---|
| 2 | $1.355 \cdot 10^{-15}$ | $4.193 \cdot 10^{-16}$ | $2.185 \cdot 10^{-4}$ | $6.500 \cdot 10^{-5}$ |
| 5 | $1.829 \cdot 10^{-15}$ | $4.897 \cdot 10^{-16}$ | $2.263 \cdot 10^{-3}$ | $1.059 \cdot 10^{-4}$ |
| 12 | $4.660 \cdot 10^{-15}$ | $7.244 \cdot 10^{-16}$ | $2.733 \cdot 10^{-2}$ | $1.897 \cdot 10^{-4}$ |
| 20 | $4.413 \cdot 10^{-15}$ | $9.160 \cdot 10^{-16}$ | 0.1239 | $2.407 \cdot 10^{-4}$ |
| 30 | $4.340 \cdot 10^{-15}$ | $9.975 \cdot 10^{-16}$ | 0.4075 | $3.115 \cdot 10^{-4}$ |
| 40 | $4.318 \cdot 10^{-15}$ | $8.591 \cdot 10^{-16}$ | 0.9669 | $3.975 \cdot 10^{-4}$ |
| 50 | $5.113 \cdot 10^{-15}$ | $9.209 \cdot 10^{-16}$ | 1.877 | $4.811 \cdot 10^{-4}$ |
| 60 | $4.761 \cdot 10^{-15}$ | $7.906 \cdot 10^{-16}$ | 3.457 | $5.751 \cdot 10^{-4}$ |
| 70 | $5.026 \cdot 10^{-15}$ | $9.868 \cdot 10^{-16}$ | 5.381 | $6.664 \cdot 10^{-4}$ |
| 80 | $5.271 \cdot 10^{-15}$ | $8.843 \cdot 10^{-16}$ | 8.633 | $7.605 \cdot 10^{-4}$ |
| 90 | $5.448 \cdot 10^{-15}$ | $9.287 \cdot 10^{-16}$ | 13.42 | $8.527 \cdot 10^{-4}$ |
| 100 | $5.091 \cdot 10^{-15}$ | $9.342 \cdot 10^{-16}$ | 19.69 | $9.484 \cdot 10^{-4}$ |

Table 1: Comparison of numerical complexity and accuracy between a traditional Gauss-Jordan solver and the banded algorithm described just before this, for the calculation of the Lagrange multipliers on a series of polyalanine chains as a function of their number of residues $N_{\text{res}}$.

In fig. 3.1 and table 3, we compare both the execution time (a) and the accuracy (b) of the two different methods: Gauss-Jordan elimination[43], and the banded recursive solution advocated here and made possible by the appropriate indexing of the constraints. The calculations have been run on a Mac OS X laptop with a 2.26 GHz Intel Core 2 Duo processor, and the errors were measured using the normalized deviation of $R\lambda$ from $-o$, i.e., if we denote by $\lambda$ the solution provided by the numerical method,

$$\text{Error} := \frac{\sum_{I=1}^{N_c} \left| \sum_{J=1}^{N_c} R_{IJ}\lambda_J + o_I \right|}{\sum_{I=1}^{N_c} |\lambda_I|} \,. \tag{3.1}$$

From the obtained results, we can see that both methods produce an error which is very small (close to machine precision), being the accuracy of the banded algorithm advocated in this work slightly higher. Regarding the computational cost, as expected, the Gauss-Jordan method presents an effort that approximately scales with the cube of the number of constraints $N_c$ (which is approximately proportional to $N_{\text{res}}$), while the banded technique allowed by the particular structure of the matrix $R$ follows a fairly accurate linear scaling. Although it is typical that, when two such different behaviours meet, there exists a range of system sizes for which the method that scales more rapidly is faster and then, at a given system size, a crossover takes place and the slower scaling method becomes more efficient from there on, in this case, and according to the results obtained, the banded technique is less time-consuming for all the explored molecules, and the crossover should exist at a very small system size (if it exists at all). This is very relevant for any potential uses of the methods introduced in this work.

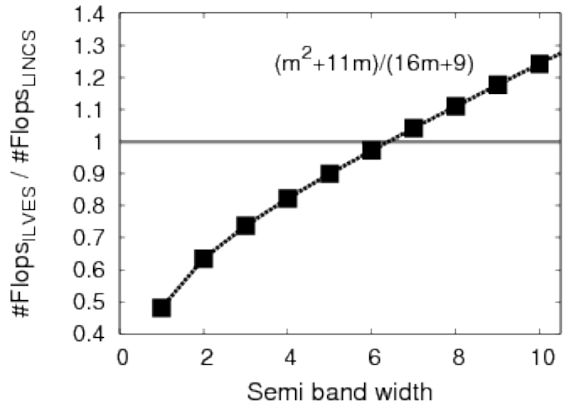


Figure 4.1: Comparison of the efficiencies to solve the linear sparse system (2.17). It is expressed as the quotient of the number of flops of the banded algorithm (ILVES, based on (2.20)) divided by that of the approximation $(I - B)^{-1} \simeq I + B + B^2 + B^3 + B^4$ (LINCS) as a function of the semi-band width $m$ of the sparse banded matrix $B$.

# 4 Conclusions

We have shown that, if we are dealing with typical biological polymers, whose covalent connectivity is that of essentially linear objects, the Lagrange multipliers that need to be computed when $N_c$ constraints are imposed on their internal degrees of freedom (such as bond lengths, bond angles, etc.) can be obtained in $O(N_c)$ steps as long as the constraints are indexed in a convenient way and banded algorithms are used to solve the associated linear system of equations. This path has been traditionally regarded as too costly in the literature [19–25], and, therefore, our showing that it can be implemented efficiently could have profound implications in the design of future molecular dynamics algorithms.

Since it is well-known that the use of the exact Lagrange multipliers to solve the underlying ordinary differential equations produces unstable algorithms [19,36], the field of imposition of constraints in molecular dynamics simulations is dominated by methods that ensure that the system stays exactly on the constrained subspace as the simulation proceeds, being some of the most popular approaches SHAKE [19], RATTLE [44] and LINCS [37]. The technique introduced in this work can be used to improve this type of methods:

- SHAKE and RATTLE consider a system of equations ($\sigma = 0$) which ensures that constraints are satisfied for every time step within a given tolerance. To this end, they iteratively solve an approximated linear problem. Although they do not explicitly calculate the exact Lagrange multipliers, but approximations to them that enforce the satisfaction of the constraints, the (non-symmetric) matrices appearing in both algorithms can be expressed as banded matrices if the constraints are cleverly ordered (see supplementary material), and therefore they can be inverted with a banded algorithm like the one described here.

- LINCS does use the same equation ($\frac{d^2\sigma}{dt^2} = 0$) as we do to calculate the Lagrange multipliers, and then adds a correction term to ensure that the constraints are satisfied in every time step. But instead of explicitly inverting $R$ in (2.6), LINCS con-

siders $B := I - R$ (being $I$ the identity matrix), and then approximates $(I - B)^{-1} \simeq I + B + B^2 + B^3 + B^4$. This assumption is very efficient, but is only valid when the absolute value of all eigenvalues of $B$ is less than 1, what in practice precludes the imposition of constraints on bond angles[37]. Our banded method can efficiently solve the system without having this drawback, thus allowing to be inserted into LINCS to constrain bond angles. We call this method ILVES (Finnish word for "lynx"). The number of flops required by ILVES and LINCS (taken as an estimation of their numerical complexities) is similar for the step of solving the linear system, as it can be seen in fig. 4.1. We calculated the number of flops as follows. For ILVES (as can be seen in[18], where the banded operations are described), getting the diagonal $\xi$'s in (2.20) takes about $2mN_c$ flops, being $m$ the semi-band width of the matrix $R$; calculating the $\xi$'s above the diagonal takes about $m^2N_c$ flops, and calculating those below the diagonal only $mN_c$ flops using (2.21). Once all $\xi$'s are known, solving the system takes about $8mN_c$ flops. Therefore, ILVES takes about $(m^2 + 11m)N_c$ flops to solve the linear banded system. If we use $(I - B)^{-1} \simeq I + B + B^2 + B^3 + B^4$ being $B$ a banded matrix with semi-band width $m$, we first have to build $B$ from $R$, what takes $N_c$ flops. Then matrix-vector products are performed, each taking about $(2m + 1)N_c$ products and $2mN_c$ additions. Since it has to be done four times, and the corresponding terms have to be added, the total number of flops will be about $(16m + 9)N_c$. The quotient between both numerical complexities increases as a function of $m$, but in the most interesting cases of biological polymers, $m$ will not be very large, as we have shown in the previous sections. Notice that, in fig. 4.1, we are not comparing the total number of operations of ILVES and LINCS, but only the part of them in which $R^{-1}o$ is computed. Building the matrix $R$ and the calculation of the subsequent correcting terms are common steps in both algorithms.

It is worth mentioning that the accuracy up to which constraints are satisfied depends on the method to implement them. SHAKE and RATTLE rely on an iterative procedure, which is repeated until every constraint is satisfied within a chosen arbitrary tolerance[19,45]. The accuracy of LINCS depends on the order used in the expansion $(I - B)^{-1} \simeq I + B + B^2 + B^3 + B^4 + \ldots$. For MD calculations a fourth order in the expansion is normally enough, while for Brownian dynamics (with longer time steps) an eighth order expansion may be necessary[46]. In any case, higher accuracies require more time-consuming calculations (this increase in the cost can be very high[47]). Therefore, an equilibrium between accuracy and numerical complexity has to be sought by an appropriate choice of the parameters present in any method (except for methods such as the one introduced here, which does not contain any parameter). To this end, prior to any simulation, some tests have to be performed. These tests (which can be run either on the system we are going to simulate or on a toy one) must measure the accuracy of a series of observable quantity as a function of the parameter contained in the method (like `TOL` in Amber's[48] SHAKE, `shake_tol` in Gromacs'[49] SHAKE or `lincs_order` in Gromacs' LINCS). With this data, together with the computational cost as a function of the same parameter, a practical decision must be taken based on available resources and desired accuracy.

It is clear from the presented examples that, in addition to its application to SHAKE, RATTLE and LINCS, the new techniques introduced here can be applied to many of the rest of approaches mentioned in sec. 1, as well as to the development of new such techniques.

Finally, we are exploring an extension of the ideas introduced here to the calculation not only of the Lagrange multipliers but also of their time derivatives, to be used in higher order integrators than Verlet. Since the sparsity of the matrices to calculate these derivatives is the same as that of the matrix $R$ in this work, the same banded techniques can be used to solve the problem.

# Acknowledgements

We would like to thank Giovanni Ciccotti for illuminating discussions and wise advice, and Claudio Cavasotto and Isaías Lans for their help with the setting up and use of AMBER. The numerical calculations have been performed at the BIFI supercomputing facilities; we thank all the staff there for their help and technical assistance. We also thank the anonymous referees for pointing out a number of issues that have significantly improved the clarity of the manuscript.

This work has been supported by the grants FIS2009-13364-C02-01 (MICINN, Spain), Grupo de Excelencia "Biocomputación y Física de Sistemas Complejos", E24/3 (Aragón region Government, Spain), ARAID and Ibercaja grant for young researchers (Spain). P. G.-R. is supported by a JAE Predoc scholarship (CSIC, Spain).

# References

[1] Echenique, P., *Contemp. Phys.*, 2007, **48**, 81–108.

[2] Cedar, H. and Bergman, Y., *Nat. Rev. Genet.*, 2009, **10**, 295–304.

[3] Piana, S.; Sarkar, K.; Lindorff-Larsen, K.; Minghao, G.; Gruebele, M. and Shaw, D. E., *J. Mol. Biol.*, 2011, **405,1**, 43–48.

[4] Shaw, D. E.; Ron O. Dror, R.; Salmon, J.; Grossman, J.; Mackenzie, K.; Bank, J.; Young, C.; Batson, B.; Bowers, K.; Edmond Chow, E.; Eastwood, M.; Ierardi, D.; John L. Klepeis, J.; Jeffrey S. Kuskin, J.; Larson, R.; Kresten Lindorff-Larsen, K.; Maragakis, P.; M.A., M.; Piana, S.; Yibing, S. and Towles, B., *In Proceedings of the ACM/IEEE Conference on Supercomputing (SC09), ACM Press, New York,*, 2009.

[5] de Jong, W. A.; Bylaska, E.; Govind, N.; Janssen, C. L.; Kowalski, K.; Mller, T.; Nielsen, I.; van Dam, H.; Veryazov, V. and Lindh, R., *PCCP*, 2010, **12**, 6896–6920.

[6] Phillips, J. C.; Braun, R.; Wei, W.; Gumbart, J.; Tajkhorshid, E.; Villa, E.; Chipot, C.; Skeel, R. D.; Kal, L. and Schulten, K., *JCC*, 2005, **26,16**, 1781–1802.

[7] Pearlman, D. A.; Case, D. A.; Caldwell, J. W.; Ross, W. R.; Cheatham III, T. E.; DeBolt, S.; Ferguson, D.; Seibel, G. and Kollman, P., *Comp. Phys. Commun.*, 1995, **91**, 1–41.

[8] Gonnet, P.; Walther, J. H. and Koumoutsakos, P., *Journal of Chemical Physics*, 2009, **180**, 360–364.

[9] Verlet, L., *Phys. Rev.*, 1967, **159**, 98–103.

[10] Frenkel, D. and Smit, B., *Understanding molecular simulations: From algorithms to applications*, Academic Press, Orlando FL, 2nd ed., 2002.

[11] Feenstra, K. A.; Hess, B. and Berendsen, H. J. C., *J. Comput. Chem.*, 1999, **20**, 786–798.

[12] Eastman, P. and Pande, V. S., *J. Chem. Theory Comput.*, 2010, **6 (2)**, 434–437.

[13] Mazur, A. K., *J. Phys. Chem. B*, 1998, **102**, 473–479.

[14] Echenique, P.; Calvo, I. and Alonso, J. L., *J. Comput. Chem.*, 2006, **27**, 1748–1755.

[15] Van Gunsteren, W. F. and Karplus, M., *Macromolecules*, 1982, **15**, 1528–1544.

[16] Barth, E.; Kuczera, K.; Leimkuhler, B. and Skeel, R. D., *J. Comput. Phys.*, 1995, **16 (10)**, 1192–1209.

[17] Goldstein, H.; Poole, C. and Safko, J., *Classical Mechanics*, Addison-Wesley, 3rd ed., 2002.

[18] García-Risueño, P. and Echenique, P., *Submitted*, 2010.

[19] Ryckaert, J. P.; Ciccotti, G. and Berendsen, H. J. C., *J. Comput. Phys.*, 1977, **23**, 327–341.

[20] Dillen, J. L. M., *J. Comput. Chem.*, 1987, **8,8**, 1099–1103.

[21] Ciccotti, G. and Ryckaert, J. P., *Comput. Phys. Rep.*, 1986, **4**, 345–392.

[22] Forester, T. R. and Smith, W., *Journal of Chemical Physics*, 1997, **19**, 102–111.

[23] Krautler, V.; Van Gunsteren, W. F. and Hunenberger, P. H., *J. Comput. Chem.*, 2001, **22**, 501–508.

[24] Gonnet, P., *J. Chem. Phys.*, 2006, **220**, 740–750.

[25] Mazars, M., *J. Phys. A: Math. Theor.*, 2007, **40, 8**, 1747–1755.

[26] Weinbach, Y. and Elber, R., *J. Comput. Phys.*, 2005, **209**, 193–206.

[27] Bailey, A. G.; Lowe, C. P. and Sutton, A. P., *Journal of Chemical Physics*, 2008, **227**, 8949–8959.

[28] Bailey, A. G. and Lowe, C. P., MILCH SHAKE: An efficient method for constraint dynamics applied to alkanes Published online ahead of print in *J. Comput. Chem.* (doi: 10.1002/jcc.21237), 2009.

[29] Mazars, M., *J. Phys. A: Math. Theor.*, 2007, **49**, 1747–1755.

[30] Grønbech-Jensen, N. and Doniach, S., *J. Comput. Chem.*, 1994, **15 (9)**, 997–1012.

[31] George, A., *SIAM J. Num. Anal.*, 1973, **10**, 345–363.

[32] Rose, D. J. In *Graph Theory and Computing*, Read, R., Ed.; Academic Press, New York, 1972.

[33] Featherstone, R., *Int. J. of Rob. Res.*, 1999, **18**, 867.

[34] Bae, D.-S. and Haug, E., *Mech. Struct. and Mach.*, 1987, **15**, 359–382.

[35] Lee, K.; Wang, Y. and Chirikjian, G., *Robotica*, 2007, **25**, 739–750.

[36] Brenan, K. and Campbell, Petzold, L., *Numerical solution of initial-value problems in differential-algebraic equations*, Elsevier Science Publishing Co., New York, 1st ed., 1989.

[37] Hess, B.; Bekker, H.; Berendsen, H. J. C. and Fraaije, J. G. E. M., *J. Comput. Chem.*, 1997, **18**, 1463–1472.

[38] Henneux, M. and Teitelboim, C., *Quantization of gauge fields*, Princeton Univesity Press, 1992.

[39] Echenique, P. and Alonso, J. L., *J. Comput. Chem.*, 2006, **27**, 1076–1087.

[40] 2010.

[41] Case, D. A.; Darden, T.; Cheatham, T. E.; Simmerling, C.; Junmei, W.; E., D. R.; Luo, R.; Merz, K. M.; Pearlman, M. A.; Crowley, M.; Walker, R.; Wei, Z.; Bing, W.; Hayik, S.; Roitberg, A.; Seabra, G.; Kim, W.; Paesani, F.; Xiongwu, W.; Brozell, S. Tsui, V.; Gohlke, H.; Lijiang, Y.; Chunhu, T.; Mongan, J.; Hornak, V.; Guanglei, C. Beroza, P.; Mathews, D. H.; Schafmeister, C.; Ross, W. S. and Kollman, P. A., *University of California: San Francisco*, 2006.

[42] Yong, D.; Chun, W.; Chowdhury, S.; Lee, M. C.; Guoming, X.; Wei, Z.; Rong, Y.; Cieplak, P.; Luo, R.; Taisung, L.; Caldwell, J.; Wang, J. and Kollman, P., *J. Comput. Chem.*, 2003, **24**, 1999–2012.

[43] Press, W. H.; Teukolsky, S. A.; Vetterling, W. T. and Flannery, B. P., *Numerical recipes. The art of scientific computing*, Cambridge University Press, New York, 3nd ed., (2007).

[44] Andersen, H. C., *J. Comput. Phys.*, 1983, **52**, 24–34.

[45] Minary, P.; Martyna, G. J. and Tuckerman, M. E., *J. Comput. Chem.*, 2003, **118, 6**, 2510–2526.

[46] Apol, E.; Apostolov, R.; Berendsen, H. J. C.; van Buuren, A.; Bjelkmar, P.; van Drunen, R.; Feenstra, A.; Groenhor, G.; Kasson, P.; Larsson, P.; Meulenhoff, P.; Murtola, T.; Pall, S.; Pronk, S.; Schulz, R.; Shirts, M.; Sijbers, A.; Tieleman, P.; Hess, B.; van der Spoel, D. and Kindahl, E., 2010.

[47] Heinz, T. N.; van Gunsteren, W. F. and Hunenberger, P. H., *J. Comput. Chem.*, 2001, **115, 3**, 1125–1136.

[48] Case, D. A.; Darden, T. A.; Cheatham III, T. E.; Simmerling, C. L.; Wang, J.; Duke, R. E.; Luo, R.; Crowley, M.; Walker, R. C.; Zhang, W.; Merz, K. M.; Wang, B.; Hayik, S.; Roitberg, A.; Seabra, G.; Kolossváry, I.; Wong, K. F.; Paesani, F.; Vanicek, J.; Wu, X.; Brozell, S. R.; Steinbrecher, T.; Gohlke, H.; Yang, L.; Tan, C.; Mongan, J.; Hornak, V.; Cui, G.; Mathews, D. H.; Seetin, M. G.; Sagui, C.; Babin, V. and Kollman, P. A., Amber 10 University of California, San Francisco, 2008.

[49] van der Spoel, D.; Lindahl, E.; Hess, B.; Groenhof, G.; Mark, A. E. and Berendsen, H. J. C., *Journal of Computational Chemistry*, 2005, **26**, 1701–1719.